



GOBIERNO DEL  
ESTADO DE MÉXICO



# TESCHA

PROYECTO FINAL “CUBO 4X4”

MATERIA: LENGUAJES DE INTERFAZ

DOCENTE: ING LADISLAO ROBERTO ALDAMA ROJANO

ALUMNO: GARCIA GALICIA JURGUEN BRANDON

GRUPO: 4601

TURNO: MATUTINO

# OBJETIVO

Se realizará un cubo de leds 4x4 cuya función será encender o apagar dependiendo del valor de bit que será la suma en binario o decimal. Se armara el cubo por pisos en el que será un total de 4 pisos y una vez hecho el circuito en el Protoboard se pasara a una placa de cobre y se enmaquetara respectivamente a sus funciones ,tambien se observaran las configuraciones **pull-up & pull-down**. En algunas luces independientes.

## MARCO TEÓRICO

### MICROCONTROLADOR

Un microcontrolador (  $\mu$ C, UC o MCU) es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica.

Un microcontrolador incluye en su interior las tres principales unidades Funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada/salida.

Algunos microcontroladores pueden utilizar palabras de cuatro bits y funcionan a velocidad de reloj con frecuencias tan bajas como 4 kHz, con un consumo de baja potencia (mW o microwatts). Por lo general, tendrá la capacidad de mantenerse a la espera de un evento como pulsar un botón o de otra interrupción; así, el consumo de energía durante el estado de reposo (reloj de la CPU y los periféricos de la mayoría) puede ser sólo de nanowatts, lo que hace que muchos de ellos sean muy adecuados para aplicaciones con batería de larga duración. Otros microcontroladores pueden servir para roles de rendimiento crítico, donde sea necesario actuar más como un procesador digital de señal (DSP), con velocidades de reloj y consumo de energía más altos.

### MPLAB v8.33

Es un editor IDE gratuito, destinado a productos de la marca Microchip. Este editor es modular, permite seleccionar los distintos microcontroladores

soportados, además de permitir la grabación de estos circuitos integrados directamente al programador. Es un programa que corre bajo Windows, Mac OS y Linux.

## **PROTEUS 8.1**

Proteus es una aplicación para la ejecución de proyectos de construcción de equipos electrónicos en todas sus etapas: diseño del esquema electrónico, programación del software, construcción de la placa de circuito impreso, simulación de todo el conjunto, depuración de errores, documentación y construcción.

## **PIC16F84A**

El PIC16F84A tiene 68 bytes de RAM y 64 bytes de EEPROM de datos, tiene 12 registros de función específica FSR. Con 8 niveles de Pila hardware y tres modos de direccionamiento: directo, indirecto y relativo. Dispone de cuatro fuentes de interrupción y 13 pines de E/S (puertoA de 5 bits y el puertoB de 8 bits), con control individual bidireccional y dispone de un Timer0/ Contador con reloj independiente y la gran ventaja dispone de Perro Guardián (WDT). Este dispositivo tiene otras particularidades que lo hacen seriamente aconsejable, algunas de estas se detallan en otros artículos de esta serie y en las hojas de características del fabricante Microchip.

El PIC16F84A tiene un contador de programa de 13 bits capaz de dirigir 8 kilobytes x 14 espacio de memoria de programa. Para el F84A, el primer 1 kilobyte x 14 (0000h-03FFh) físicamente es implementado. El tener acceso a una posición anterior de la dirección físicamente puesta en práctica causará un recirculado. Por ejemplo, para posiciones 20h, 420h, 820h, C20h, 1020h, 1420h, 1820h, y 1C20h, la instrucción será la misma. El vector RESET (REPUESTO) está en la dirección 0000h y el vector INTERRUPT está en la 0004h, con cuatro fuentes de interrupción, Hay dos bloques de memoria en el PIC16F84A, estos son la memoria de programa y la memoria de datos. Cada bloque tiene su propio bus, de modo que el acceso a cada bloque pueda ocurrir durante el mismo ciclo de oscilador.



La memoria de datos adicional mayor puede ser direccionada en la RAM de objetivo general y los Registros de Función Especiales (SFRs). La operación de los SFRs que controla el “corazón” (reloj).

## INSTRUCCIONES DEL MPLAB PARA EL PIC16F84A

Hay solo 35 instrucciones en el PIC16F84A, con códigos de instrucción de 14 bits de ancho. Todas las instrucciones ocupan una palabra y todas consumen un ciclo, excepto las de salto o bifurcación que usan dos. La velocidad máxima de funcionamiento 20MHz (200 ns x instrucción). Típicamente a 4MHz (1us x instrucción), con 1024 palabras (14 bits) de memoria de programa FLASH.

- **ADDLW** Esto significa: Agregar (sumar) el Literal al registro W (acumulador o registro de trabajo) resultado en W.
- **ADDWF** Esto significa: Suma aritmética de W y un archivo (f).
- **BCF** Esto significa: Bit Clear File (pon a “0” o aclara el bit indicado (detrás de la coma) en el archivo f ). Ver también **BSF**.
- **BSF** Esto significa: Bit Set File (poner a 1 el bit indicado, en el archivo f). Ver también **BCF**.
- **BTFSC** Esto significa: Bit Test, Skip if Clear ( Bit de Test, Salta si es “0”).
- **BTFSS** Esto significa: Bit Test, Skip if Set (Bit de Test, Salta si es “1”).
- **CLRF** Esto significa: Clear f [Limpia f] (poner a 0 los 8 bits del archivo f)
- **CLRW** Esto significa: Clear **W** (limpiar el registro de trabajo).
- **CLRW** El registro W es aclarado, todos los bits son puestos a 0.
- **DECF** Esto significa: Decremento del archivo f .
- **GOTO** Esto significa: Bifurcación Incondicional.
- **INCF** Esto significa: Incrementar el archivo f.
- **MOVFW** Esta forma de instrucción, no es válida (**no se recomienda su uso**), a pesar de que el propio **MPLAB** la admita, significa mover el contenido del archivo **F**, al registro de trabajo **W**.



- **RETURN** Esto significa: Retorno de Subrutina. Esta instrucción está al final de una rutina o subrutina. No afecta al registro STATUS.
- **MOVLW** Esto significa: Mueve Literal a W.
- **MOVWF** Esto significa: Copia **W** al archivo llamado **f**.

## LISTA DE MATERIALES

- **2 Protoboard**
- **2 Capacitores cerámicos de 22 mf**
- **1 Cristal de 4 MHz**
- **22 Leds**
- **4 Resistencias de 470 Ohms**
- **4 Resistencias de 10 k**
- **8 Resistencias de 220 Ohms**
- **8 Transistores 2N2222**
- **Cables Jumper**
- **3 Push Button**
- **Proteus 8.3**
- **MpLab v8.33**

# DESARROLLO

## CODIGO

1. Crear un nuevo archivo con extensión .ASM y nombre cualquiera
2. Crear un Proyecto nuevo eligiendo un nombre y ubicación
3. Agregar el archivo .ASM como un SOURCE FILE
4. Elegir el microcontrolador a utilizar desde SELECT DEVICE del menú CONFIGURE
5. Ingresar las librerías para la instrucción de retardos INCLUDE RETARDOS

```
LIST F=16F84A
INCLUDE <P16F84A.INC>
__CONFIG __CP_OFF & __WDT_OFF & __PWRTE_ON & __XT_OSC
CBLOCK 0x0c

ENDC

ORG 0

BSF STATUS,RPC ; BIT EN UNO
CLRF TRISE ;SALIDA
BCF STATUS,RPC ;INGRESO AL BANCO 0

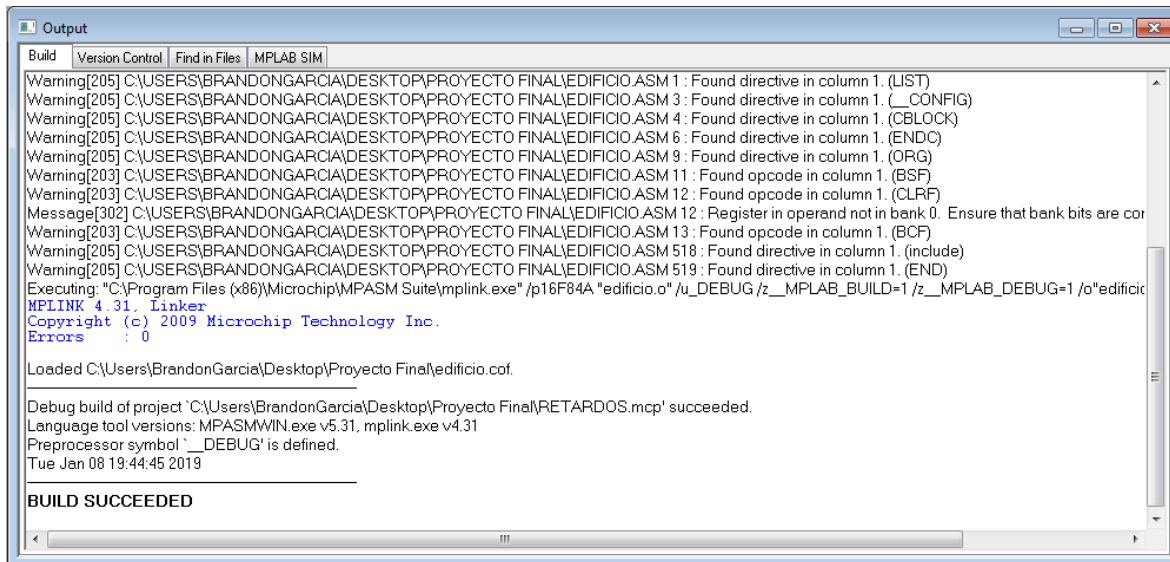
INICIC ;PRIMERA FUNCION

MOVW D'129'
MOVWF PORTB
call Retardo_500ms
MOVW D'130'
MOVWF PORTB
call Retardo_500ms
MOVW D'132'
MOVWF PORTB
call Retardo_500ms
MOVW D'136'
MOVWF PORTB
call Retardo_500ms
MOVW D'65'
MOVWF PORTB
call Retardo_500ms
MOVW D'66'
MOVWF PORTB
call Retardo_500ms
MOVW D'68'
MOVWF PORTB
call Retardo_500ms
MOVW D'72'
MOVWF PORTB

call Retardo_500ms
MOVW D'136'
MOVWF PORTB
call Retardo_500ms
MOVW D'65'
MOVWF PORTB
call Retardo_500ms
MOVW D'66'
MOVWF PORTB
call Retardo_500ms
MOVW D'68'
MOVWF PORTB
call Retardo_500ms
MOVW D'72'
MOVWF PORTB
call Retardo_500ms
MOVW D'33'
MOVWF PORTB
call Retardo_500ms
MOVW D'34'
MOVWF PORTB
call Retardo_500ms
MOVW D'38'
MOVWF PORTB
call Retardo_500ms
MOVW D'40'
MOVWF PORTB
call Retardo_500ms
MOVW D'17'
MOVWF PORTB
call Retardo_500ms
MOVW D'18'
MOVWF PORTB
call Retardo_500ms
MOVW D'20'
MOVWF PORTB
call Retardo_500ms
MOVW D'24'
MOVWF PORTB
call Retardo_500ms
```



Una vez escrito y depurado el programa, se procede a la compilación. Para esto, desde el menú PROJECT se elige la opción BUILD ALL (construir todo) que, si no existen errores, devolverá un mensaje como BUILD SUCCESSFUL.



```
Output
Build Version Control Find in Files MPLAB SIM
Warning[205] C:\USERS\BRANDONGARCIA\DESKTOP\PROYECTO FINAL\EDIFICIO.ASM 1 : Found directive in column 1. (LIST)
Warning[205] C:\USERS\BRANDONGARCIA\DESKTOP\PROYECTO FINAL\EDIFICIO.ASM 3 : Found directive in column 1. (__CONFIG)
Warning[205] C:\USERS\BRANDONGARCIA\DESKTOP\PROYECTO FINAL\EDIFICIO.ASM 4 : Found directive in column 1. (CBLOCK)
Warning[205] C:\USERS\BRANDONGARCIA\DESKTOP\PROYECTO FINAL\EDIFICIO.ASM 6 : Found directive in column 1. (ENDC)
Warning[205] C:\USERS\BRANDONGARCIA\DESKTOP\PROYECTO FINAL\EDIFICIO.ASM 9 : Found directive in column 1. (ORG)
Warning[203] C:\USERS\BRANDONGARCIA\DESKTOP\PROYECTO FINAL\EDIFICIO.ASM 11 : Found opcode in column 1. (BSF)
Warning[203] C:\USERS\BRANDONGARCIA\DESKTOP\PROYECTO FINAL\EDIFICIO.ASM 12 : Found opcode in column 1. (CLRF)
Message[302] C:\USERS\BRANDONGARCIA\DESKTOP\PROYECTO FINAL\EDIFICIO.ASM 12 : Register in operand not in bank 0. Ensure that bank bits are cor
Warning[203] C:\USERS\BRANDONGARCIA\DESKTOP\PROYECTO FINAL\EDIFICIO.ASM 13 : Found opcode in column 1. (BCF)
Warning[205] C:\USERS\BRANDONGARCIA\DESKTOP\PROYECTO FINAL\EDIFICIO.ASM 518 : Found directive in column 1. (include)
Warning[205] C:\USERS\BRANDONGARCIA\DESKTOP\PROYECTO FINAL\EDIFICIO.ASM 519 : Found directive in column 1. (END)
Executing: "C:\Program Files (x86)\Microchip\MPASM Suite\mplink.exe" /p16F84A "edificio.o" /u_DEBUG /z_MPLAB_BUILD=1 /z_MPLAB_DEBUG=1 /o"edificio
MPLINK 4.31. Linker
Copyright (c) 2009 Microchip Technology Inc.
Errors : 0

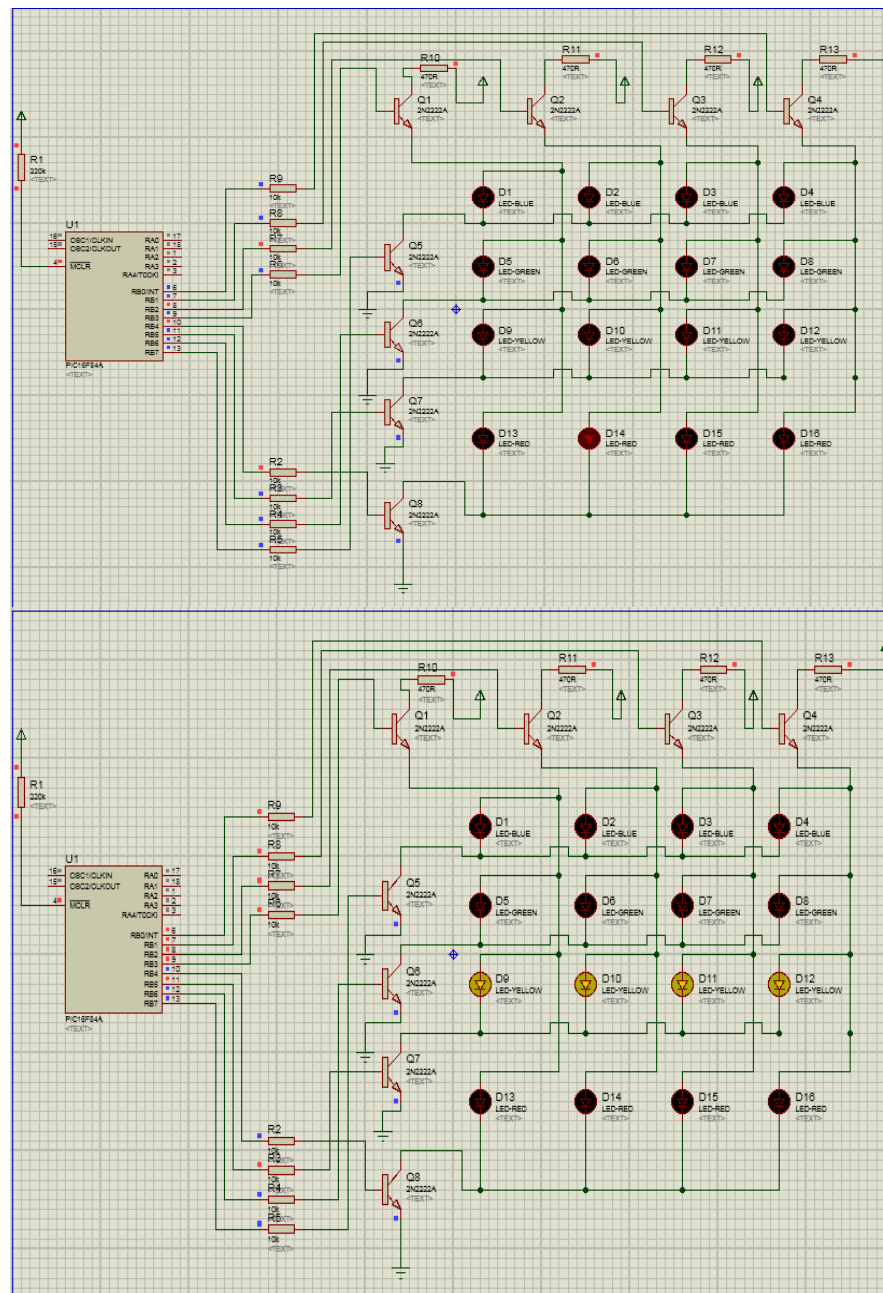
Loaded C:\Users\BrandonGarcia\Desktop\Proyecto Final\edificio.cof.

Debug build of project 'C:\Users\BrandonGarcia\Desktop\Proyecto Final\RETARDOS.mcp' succeeded.
Language tool versions: MPASMWIN.exe v5.31; mplink.exe v4.31
Preprocessor symbol '___DEBUG' is defined.
Tue Jan 08 19:44:45 2019

BUILD SUCCEEDED
```

## SIMULACION

Terminada la compilación el MPLAB® nos genera un archivo de extensión .hex el cual es completamente entendible para el PIC. Es decir, solo resta grabarlo al PIC por medio de una interfaz como por ejemplo el programador Proteus. Una vez completado esto, se alimenta al mismo y el programa ya se estará ejecutando.

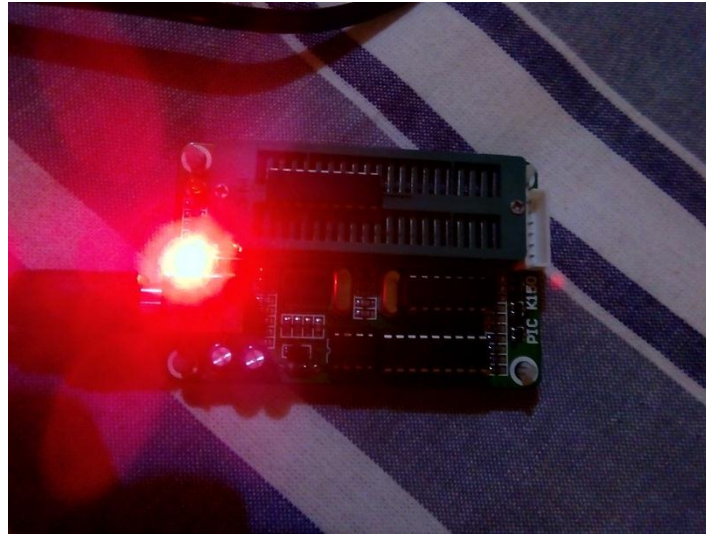




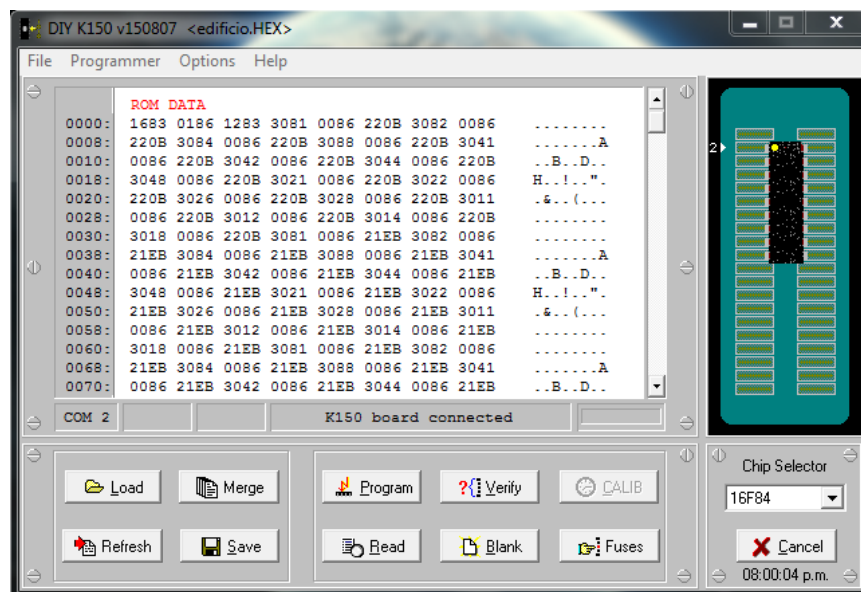
## GRABAR EL PIC 16F84A

El programa es cargado en el microcontrolador PIC16F84A, con el grabador de pic's después de ser probado en proteus.

1. Insertamos el pic en el grabador de pic's y conectamos a la Computadora.



2. Cuando ya ha leído el pic cargamos el programa primero seleccionamos borrar para verificar que no contenga nada después seleccionamos escribir y cargamos el programa.





## ARMADO FISICO

Aquí conectamos todos los elementos en el protoboar en cada entrada del pic el cristal que va conectado a al pin 15 y 16 donde salen 2 capacitores que van a tierra, el pin 4 conectamos una resistencia que va a corriente, el pin 5 está conectado a la tierra, las salidas seran las del puerto B a las que son de **RB0 a RB7**.

PROYECTO FINAL

Puerto B "Salida" Pic 16F84A

7	6	5	4	3	2	1	0
Piso 4	Piso 3	Piso 2	Piso 1	Led 4	Led 3	Led 2	Led 1
128	64	32	16	8	4	2	1
RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0

128 → + → 1 = 129 (Piso 4, Led 1)

RAZ	RA0	OSC1/CLK1	OSC2/CLK2	VDD	RB7	RB6	RB5	RB4
RAZ	RA0	OSC1/CLK1	OSC2/CLK2	VDD	RB7	RB6	RB5	RB4

DECIMAL	129	Piso 4	Led 1	RB0	Piso 1	17	Piso 1	Led 1
Piso 4	136	Piso 4	Led 2	RB1	Piso 1	18	Piso 1	Led 2
RB7	132	Piso 4	Led 3	RB2	Piso 1	20	Piso 1	Led 3
	136	Piso 4	Led 4	RB3	Piso 1	24	Piso 1	Led 4
	129	Piso 4	Led 1	RB0	Piso 1	17	Piso 1	Led 1
Piso 3	65	Piso 3	Led 1					
	66	Piso 3	Led 2					
	68	Piso 3	Led 3					
	72	Piso 3	Led 4					
	65	Piso 3	Led 1					
Piso 2	33	Piso 2	Led 1					
	34	Piso 2	Led 2					
	38	Piso 2	Led 3					
	42	Piso 2	Led 4					

En esta imagen se muestran los respectivos rangos de los valores a sumar o declarar.



## CONEXIÓN DE CUBO A PROTOBOARD

- Conectar a corriente el pin 4 por medio de una resistencia de 220kohms y 14 a corriente.
- Conectar a tierra el pin 5.
- En el pin 15 y 16 conectar un cristal de 4Mhz de las salidas conectar 2 capacitores cerámicos de 22 mf a tierra.
- En el puerto b serán las salidas en el que serán los siguientes

**RB4 – Pin10**

**RB5 – Pin 11**

**RB6 – Pin 12**

**RB7 – Pin 13**

A Cada salida se conectara una resistencia de 10K y de la salida se conectara a la base de los transistores 2N222. El Emisor será conectado a tierra correspondientemente y el colector será conectado a el número de pisos. (1, 2, 3, 4).

- De las salidas de los Pines :

**RB0 - Pin6**

**RB1 – Pin7**

**RB2 –Pin8**

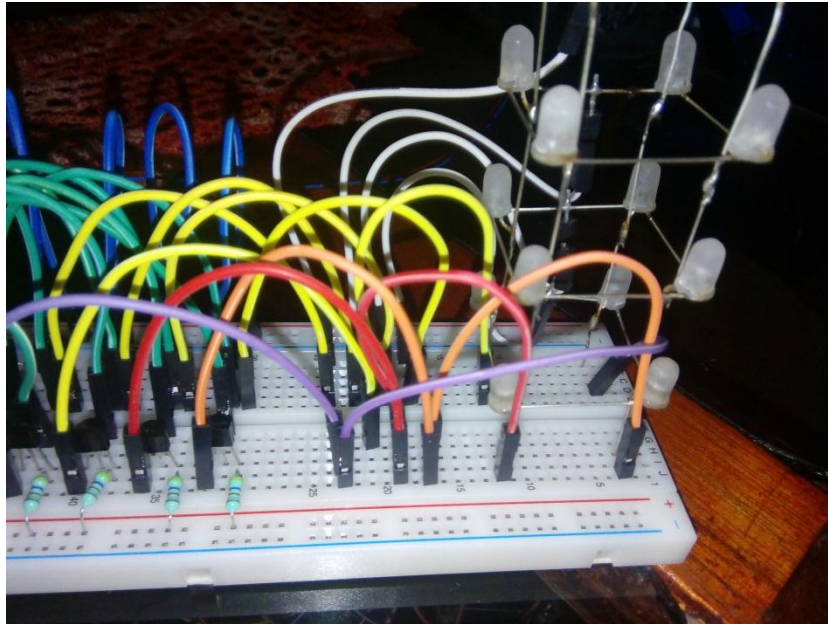
**RB3 – Pin9**

Una resistencia de 10k y en las salidas de cada una de ellas serán conectados a la base correspondientemente de los transistores 2N222, el Emisor de estos mismos serán las columnas de el cubo. (A, B, C, D). Y el colector será conectado a tierra.

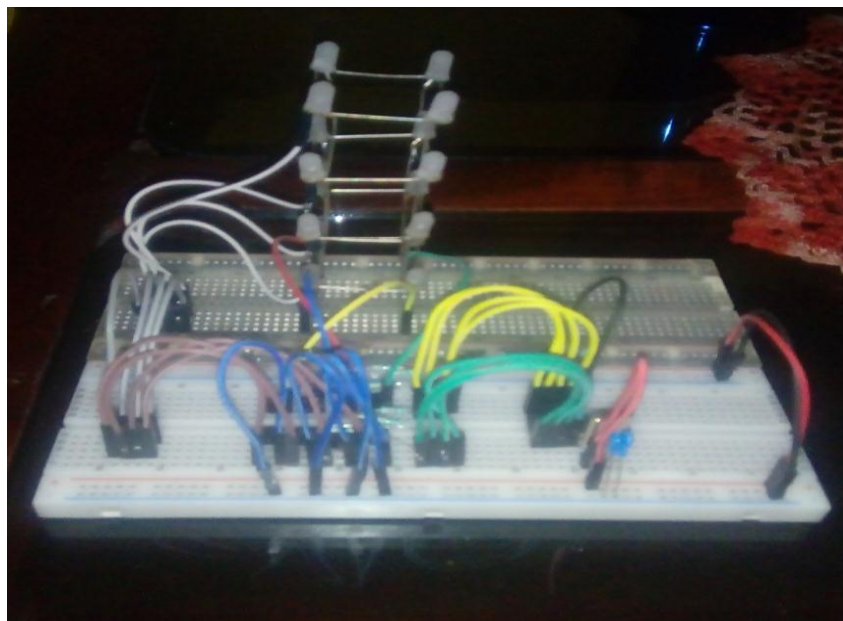
- Por otra parte serán puenteados las dos bandas de la protoboard (negativo + negativo & positivo +positivo).



En la siguiente imagen se muestran las conexiones respectivas y conectado para encender el cubo armado en el protoboard una vez finalizado.



En la siguiente imagen se puede observar que y las bandas de Conexión de polaridad ya están puenteadas. Y ensamblado el cubo en otra placa protoboard.





## ENMAQUETADO

En esta imagen se puede observar que el circuito ya fue montado en una maqueta y realizado el pintado. De igual forma el circuito esta por debado de la caja.



A continuación se observa el funcionamiento del cubo y los postes de luz conectados el paralelo y controlados en una función **Pull – Up**.

Y por otra parte una función **Pull Down** conectadas a las casas.

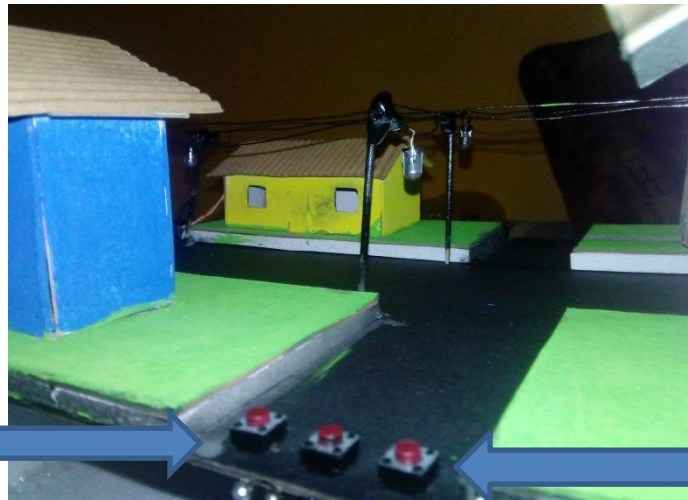




En la imagen se observan tres push button y su funcionalidad de los 3 son las siguientes:

- Push Button 1 realiza el apagado de las luces de los postes.
- Push Button 2 realiza la Función de un Reset.
- Push Button 3 realiza la función de encender los leds de las casas.

Postes de Luz



Casas

Reset



Vista de La maqueta y del cubo de leds en el que se guardo en un edificio de 4 pisos como se muestra en la imagen.

## CONCLUSION

En esta práctica tuve que realizar un programa en el debí de declarar unas instrucciones para hacer funcionar mi circuito ya que para estos antes de crear nuestro código realizamos nuestra tabla de código binario para saber cómo formaremos las cantidades de los respectivos bit (leds) a los que corresponden terminada la tabla creamos nuestro código en (MPLAB), realizamos nuestra simulación en (PROTEUS) y así grabamos nuestro pic antes de realizarlo en físico en conclusion estos lenguajes ensambladores son de faciles de manipular siempre y cuando tengamos un cierto conocimiento de este.

Por otra parte aprendemos a usar otros programas de para simulación y de lenguaje binario (maquina) ya que ofrece una satisfactoria manipulacion gracias a su interfaz gráfica.